WEST

Help | Logout

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers

Search Results -

Terms	Documents
data adj object$1	4651

Database: US Patents Full-Text Database

Refine Search: data adj object$1

Search History

DB Name	Query	Hit Count	Set Name
USPT	data adj object$1	4651	L19
USPT	data adj object$1	4651	L18
USPT	l16 same query same definition$1	4	L17
USPT	meta adj data	209	L16
USPT	l13 or l12 or l11 or l14	2017	L15
USPT	l4 or l5 or l6 or l7 or l8 or l9 or l10	1364	L14
USPT	709/219.ccls.	405	L13
USPT	709/218.ccls.	227	L12
USPT	709/217.ccls.	406	L11
USPT	709/202.ccls.	189	L10
USPT	709/201.ccls.	316	L9
USPT	709/203.ccls.	521	L8
USPT	709/200.ccls.	209	L7
USPT	709/317.ccls.	48	L6
USPT	709/316.ccls.	69	L5
USPT	709/315.ccls.	200	L4
USPT	object adj type$1	2752	L3
USPT	meta adj data adj service	1	L2
USPT	meta adj object	13	L1

Document Number 7

Entry 7 of 13 File: USPT Feb 10, 1998

DOCUMENT-IDENTIFIER: US 5717925 A
TITLE: Information catalog system with object-dependent functionality

DEPR:
The provision and categorization of cataloging functions by the database catalog system 26 may be accomplished in several ways by persons skilled in the art based on the disclosure herein. The system 26 is preferably implemented by a general purpose computer, such as the data processing node 4 of FIG. 1, which executes a sequence of processing commands provided by a database catalog computer program. The program could be written in a variety of computer languages including the well known "C" programming language. In a preferred embodiment, the cataloging functions of the database catalog system 26 are implemented as a series of C language functions. Those functions preferably provide sort-query-logic (SQL) database access capability. As described in more detail below, the object types and object instances are generated as data structures having a plurality of defined fields corresponding to contiguous memory locations that store the information which defines the object types and the instances thereof. One of the data structure fields identifies the functional category of which the object type or object instance is a member. When a call is made to one of the cataloging functions that operate on object types and instances, the procedure reads the category identifier of the object type or instance data structure. If the category identifier is recognized, the function is executed. If not, an error message is displayed advising that the requested procedure is not available for the object type or object instance identified. In this way, the meta objects of the database catalog system 26 encapsulate the functions and properties which they inherit from the function category classes to which they belong. Such encapsulation could be similarly achieved using object oriented programming tools such as the C++ programming language.

DEPL:
META OBJECT DATA STRUCTURES

Document Number 7

Entry 7 of 13 File: USPT Feb 10, 1998

US-PAT-NO: 5717925
DOCUMENT-IDENTIFIER: US 5717925 A
TITLE: Information catalog system with object-dependent functionality
DATE-ISSUED: February 10, 1998
US-CL-CURRENT: 707/102; 707/103
APPL-NO: 8/ 658402
DATE FILED: June 5, 1996
PARENT-CASE:
This application is a continuation of application Ser. No. 08/134,355, filed Oct. 8, 1993, now abandoned.

Document Number 1

Entry 1 of 1 File: USPT Oct 5, 1999

DOCUMENT-IDENTIFIER: US 5963958 A
TITLE: Method and system for generating object definitions

DEPR:
The method then proceeds to step 106, where database interface 35 queries database 28 regarding the object class or instance identified in step 104. For example, if database 28 is created using the Unisys Universal Repository.TM. DBMS 30, then database interface 35 queries a meta data services function of DBMS 30 using the name of the selected object class or instance. In response, the meta data services function returns a list of variables, methods and/or subclasses associated with the selected object class or instance. This information is received by database interface 35 at step 108.

DEPR:
If the object information received at step 108 does include one or more methods, then at step 112, database interface 35 queries database 28 regarding the first method identified in step 108. For example, if DBMS 30 is a Unisys Universal Repository.TM., then database interface 35 queries the meta data services function of DBMS 30 using the name of the selected method. In response, the meta data services function returns a set of source code for the selected method. The source code, which is written in the language of development environment 24, in this example Smalltalk, is received by database interface 35 at step 114.

Document Number 1

Entry 1 of 1 File: USPT Oct 5, 1999

US-PAT-NO: 5963958
DOCUMENT-IDENTIFIER: US 5963958 A
TITLE: Method and system for generating object definitions
DATE-ISSUED: October 5, 1999
US-CL-CURRENT: 707/104; 707/100
APPL-NO: 8/ 897081
DATE FILED: July 18, 1997

Main Menu | Search Form | Result Set | ShowS Numbers | Edit S Numbers | Referring Patents
First Hit | Previous Document | Next Document
Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

Help Logout

Document Number 2

Entry 2 of 4 File: USPT Jan 4, 2000

DOCUMENT-IDENTIFIER: US 6012067 A
TITLE: Method and apparatus for storing and manipulating objects in a plurality of relational data managers on the web

DEPR:
In this invention, a logical relational database schema is partitioned over many component schema. One logical schema distributes over many physical schema over the net. A SQL (Structured Query Language) query can be triggered against the logical schema without worrying about the physical distribution of the components. FIG. 10 shows a block diagram of a single logical relational database schema with business logic. It is partitioned over schema 1, schema 2 etc. As mentioned earlier, each such component schema consists of packages implementing relevant portion of the business logic for the data stored in that component database. Component schema and packages talk to each other through object request brokers. A query with table names, attribute names, method names from various component schema is resolved by successive preparations and collaborative executions at different sites over the internet. In practice, SQL query is resolved by first parsing and then executing relational operations over the data stored in tables. Parsing phase consists of recognizing table, attribute and package definitions stored in the data dictionary. This information stored in data dictionary is often called the meta data. This invention incorporates a preparation phase and an execution phase for each query. During the preparation phase, a query is parsed at each location to find whether the meta data stored here is sufficient to fully resolve the query. If not, then the unresolved portion is sent to the appropriate remote site. This phase includes loading of packages for user defined types along with necessary initializations. The second phase is the phase of execution where queries are executed, joins are performed and business logic is applied. Application of specific business logic may not be complete unless another embedded program finishes execution and sends the result back from another site. This results in collaborative executions across various sites. A preparation phase followed by an execution phase for SQL queries over disparate schema components presented in this invention is unique and not present in current relational database products.

CLPV:
B) Parsing a Structured Query Language (SQL) query at a physical component schema location to resolve definitions for tables, types and packages whatever is locally available and extracting portion of the query for sending it to other locations for preparation wherever the relevant meta data is available;

CLPV:
C) Preparing fully a SQL query at multiple sites by successively parsing with locally available meta data and initializing any object packages used in attribute definitions in tables at each component schema location;

Document Number 2

Entry 2 of 4 File: USPT Jan 4, 2000

US-PAT-NO: 6012067
DOCUMENT-IDENTIFIER: US 6012067 A
TITLE: Method and apparatus for storing and manipulating objects in a plurality of relational data managers on the web
DATE-ISSUED: January 4, 2000
US-CL-CURRENT: 707/103; 703/3, 703/4, 707/10, 707/104, 707/201, 709/203, 709/229, 713/200, 713/201
APPL-NO: 9/ 033325
DATE FILED: March 2, 1998

Main Menu | Search Form | Result Set | ShowS Numbers | Edit S Numbers | Referring Patents
First Hit | Previous Document | Next Document
Full | Title | Citation | Front | Review | Classification | Date | Reference | Claims | KWIC

Help Logout

Document Number 2

Entry 2 of 13 File: USPT Nov 23, 1999

DOCUMENT-IDENTIFIER: US 5991823 A
TITLE: Low overhead object adaptor

DRPR:
FIG. 2 shows a subcontract registry table having various subcontract meta objects as entries according to one embodiment of the present invention.

DEPR:
The subcontract registry 200 includes a Subcontract Identifier column 202, an associated quality of service list column 204, a subcontract client representation create function column 206, and location indicators to other functions 208. Each row of the table 210 is termed a Subcontract Meta Object and, by way of example, may be implemented as a C++ object. In the embodiment shown, a plurality of Subcontract Meta Objects 212, 214 and 216 are provided in the subcontract registry 200. The first Subcontract Meta Object 212 has a Subcontract Identifier of "1" and is thus identified as Subcontract 1. Subcontract 1 lists the following features for its quality of service: clean shutdown, security, persistence and server activation. The name-value pairs in this quality of service list indicates that a clean shutdown will not be implemented, that for security, an authentication protocol using MD5 will be used, that persistence is turned on and that server activation is present. In column 206, Subcontract 1 has a location indicator to its associated subcontract client representation create function, Client Rep Create 1. Column 208 includes a plurality of location indicators to various other functions associated with this Subcontract Meta Object such as location indicators to an unmarshal function, a destringify function and a bad server identifier handler.

DEPR:
The second illustrated Subcontract Meta Object 214 is the Subcontract Meta Object for Subcontract 2. The quality of service list for this subcontract indicates that a clean shutdown will be implemented, that for security, an authentication protocol using MD5 will be used, that persistence is turned on and that server activation is present. The Subcontract Meta Object 216 indicates that for Subcontract 3 it will allow clean shutdowns, that for security, an authentication protocol using MD5 will be used, that persistence is present and that server activation will also be turned on.

DEPR:
The subcontract registry 200 will typically have a group of associated functions that are used to organize and access the registry. By way of example, the associated functions may include an Add function, a Find function, a Get First function and a Get Next function. The Add function may be used to add a new quality of service to the table. In the described embodiment, the Add function takes as arguments a Subcontract Identifier and a Subcontract Meta Object. A Find function takes as an argument a Subcontract Identifier and returns the Subcontract Meta Object associated with that identifier. The functions Get First and Get Next return the appropriate Subcontract Meta Object and are used to iterate over the entire table and thus search it completely for a particular quality of service. This subcontract registry may be used in the following manner. When a client wishes to make a call to a particular server object, the subcontract registry may be used to look up the Subcontract Identifier associated with that server object and then to call the appropriate subcontract client representation create function in order to create an object reference to the particular server object using the appropriate features.

DEPR:

Each implementation definition represents an entry in the implementation registry that contains location indicators to the different data stored. The implementation registry 250 includes an Implementation Identifier column 252 that names the implementation, a Subcontract Meta Object column 254, an Interface Identifier column 256, a column 257 for a Ready Flag, a call back functions column 258, and a skeleton information column 259. The Implementation Identifier is a name for the implementation that is supplied by the developer when an implementation definition is created. The Subcontract Meta Object is a location indicator from a particular implementation definition to a Subcontract Meta Object contained in the subcontract registry 200. The Interface Identifier is a fixed globally unique name of the type of a particular interface. The Ready Flag will be set for a particular implementation definition when the implementation has been prepared as will be described below with reference to FIG. 8. If the Ready Flag is not set, then the dispatch function may have to wait temporarily until the implementation is ready, as described below with reference to FIGS. 12a and 12b, and specifically in step 727 of FIG. 12b. The skeleton information provides information and functions for use by the skeleton associated with this implementation. The call back functions are a set of functions associated with each implementation. A wide variety of call back functions may be associated with a particular implementation. By way of example, the call back functions Lookup, Post Invoke, Revoke, Deactivate and Shutdown will be illustrated.

DEPR:
FIG. 4 at 300 shows how the subcontract registry 200 and the implementation registry 250 interact with one another. The subcontract registry 200 registers various subcontracts by having Subcontract Meta Object entries such as 212, 214 and 216, etc. Each of these Subcontract Meta Objects identifies a unique set of features of the system. For example, Subcontract Meta Object 1 identifies Subcontract 1 that identifies the features security 302, persistence 304 and clean shutdown 306. Similarly, Subcontract 2 identifies the feature server activation 308. Subcontract 3 identifies the features clean shutdown 306, server activation 308 and transactions 310.

DEPR:
The interaction between the two takes place as follows. The implementation registry 250 identifies various implementation definitions such as PRINTER 262 and MODEM 264. Each of these implementation definitions references a single one of the Subcontract Meta Objects through, for example, link 266. It may be possible for a particular Subcontract Meta Object to be referenced by more than one implementation definition. For example, both the PRINTER and the MODEM definitions reference Subcontract 1 through Subcontract Meta Object 1.

DEPR:
FIG. 5 at 150 shows the object reference described above in the overview. It should also be noted that the subcontract Identifier 158 identifies not only which subcontract the object will utilize but also identifies a particular Subcontract Meta Object in the subcontract registry through column 202 of FIG. 2. The Implementation Identifier 162 identifies the name of the implementation for this object and also indicates an implementation definition by way of column 252 of the implementation registry of FIG. 3.

DEPR:
Referring next to FIG. 7, a create implementation definition function suitable for implementing step 402 in FIG. 6 will be described in more detail. The create implementation definition function takes as arguments a quality of service list, a name for an implementation and an Interface Identifier. In step 452 the desired quality of service list is used to search through the subcontract registry and match the quality of service list of an existing Subcontract Meta Object. This step may be performed by searching each entry in the subcontract registry using the subcontract registry functions as discussed above. In step 454 an entry is created in the implementation registry using the Interface Identifier and the name for the implementation as the Implementation Identifier. In step 456 the Subcontract Meta Object field for this new entry is updated to point to the identified Subcontract Meta Object found in step 452. Next, in step 457 all skeleton information is stored in this new entry, including the skeleton dispatch function unique for this implementation definition. After this step this new implementation definition is returned and this function is done and control returns to step 404 of FIG. 6.

DEPR:
Referring next to FIG. 9, a create object reference function suitable for implementing step 406 in FIG. 6 will be described in more detail. This function takes as arguments an implementation definition and a User Key. It will

eventually create and return an object reference by using the subcontract client representation create function described below. In step 502 the implementation definition is used to reference its corresponding entry in the implementation registry in order to produce the corresponding Subcontract Meta Object location indicator that points to the appropriate Subcontract Meta Object in the subcontract registry. In step 504 the subcontract client representation create function corresponding to the entry in the table for this found Subcontract Meta Object is returned. In step 506 this subcontract client representation create function is called with the User Key and the received implementation definition as arguments. This client representation create function creates an object reference for a servant (that may or may not yet exist) corresponding to the Interface Identifier and named Implementation Identifier of the received implementation definition. Because this client representation create function is unique to each subcontract, this step utilizes the appropriate features of the corresponding subcontract in order to return the object reference. In step 508 this object reference created is returned and the function is done and control returns to step 408 of FIG. 6.

DEPR:
In step 706 the server identifier is extracted from the object reference in the marshal buffer. The server object that is the subject of the invocation is present within a particular server process on a host computer; thus, it is important to verify that this server identifier matches with the identifier of the current server. In step 708 this extracted server identifier is compared with the identifier of the current server in order to determine if the object reference is referencing the appropriate server process. Step 710 determines if the extracted server identifier is valid and does match with the current server. If not, this indicates that the server identifier is invalid and appropriate action should be taken. This action may be performed by a bad server identifier handler function. Step 712 determines if an appropriate bad server identifier handler is registered in the subcontract meta object. Because the subcontract identifier has already been extracted from the object reference, this step may be performed by using the subcontract registry of FIG. 2. The subcontract identifier acts as a key to a particular row of this table that allows a search of column 208 in order to determine if the bad server identifier handler is present. If the handler is not present, then the system throws an exception relating to this situation in step 716 and the dispatch function ends. If, however, the handler is registered in the subcontract meta object, then in step 714 this bad server identifier handler is called with the subcontract identifier and the marshal buffer as arguments. After this step, the dispatch function ends.

Document Number 2

Entry 2 of 13 File: USPT Nov 23, 1999

US-PAT-NO: 5991823
DOCUMENT-IDENTIFIER: US 5991823 A
TITLE: Low overhead object adaptor
DATE-ISSUED: November 23, 1999
US-CL-CURRENT: 709/330
APPL-NO: 8/ 669782
DATE FILED: June 26, 1996
PARENT-CASE:
CROSS REFERENCE TO RELATED APPLICATIONS U.S. patent application "Method and Apparatus for Subcontracts in Distributed Processing Systems", Ser. No. 08/554,794 filed Nov. 7, 1995, a continuation of Ser. No. 07/995,863, filed Dec. 21, 1992, now abandoned, is related to the present application and is incorporated by reference herein in its entirety. Additionally, the following U.S. patent applications, all filed concurrently herewith, are related to the present application and are incorporated by reference herein in their entirety: Ser. No. 08/670,684, Ser. No. 08/673,181, Ser. No. 08/670,681, Ser. No. 08/670,700, Ser. No. 08/670,682.